Capítulo 4

IMPLEMENTACIÓN EN MATLAB DEL MÉTODO LIVE WIRE

En este capítulo se explicará el funcionamiento del programa realizado en MATLAB que implementa el método de segmentación Live Wire tratado en el Capítulo 3. Se analizará de qué forma interviene el usuario y cómo puede condicionar resultados finales, y se analizará el formato y estructura que tiene.

Por último, se detalla una breve descripción de todas las funciones que utiliza el programa, además de dar una visión general de dónde y bajo qué condiciones actúan éstas con la inclusión de un pseudocódigo que facilita la total comprensión del programa.

4.1. EXPLICACIÓN GENERAL DEL PROGRAMA

El programa de segmentación realizado en MATLAB se llama '*seg*'. Su código, como el de resto de funciones, se pueden consultar en el Apéndice A del presente documento. El objetivo de *seg* es el de segmentar una imagen, de manera que vamos a detallar el formato que tiene para poder así dominar las opciones que contiene y los resultados que nos proporciona.

Recibe cuatro parámetros de entrada. El más importante, obviamente, será la imagen que se quiere segmentar, el resto de parámetros, o son opciones dadas para la segmentación que el usuario debe conocer para obtener un mejor rendimiento, u opciones de presentación de resultados. Como salida no devuelve ningún parámetro, pero sí crea dos archivos que mostrarán los resultados obtenidos tras la segmentación. De todas formas, el proceso de segmentación es visualizado en todo momento por la pantalla del ordenador, de manera que si la segmentación obtenida no es satisfactoria, el usuario pueda cancelar el proceso y reiniciarlo en cualquier momento.

La línea de comandos de *seg* es la siguiente:

```
seg (filename, margen, flag_marco, color_option)
```

siendo

filename: ruta del archivo de la imagen que se desea segmentar.

margen: entero que determina el margen que habrá en el proceso de segmentación.

flag_marco: bandera para indicar si se desea o no que el marco sea un camino mínimo.

color_option: opción para escoger los colores con los que mostrar la segmentación.

Todos ellos se explican a continuación.

4.1.1. ENTRADAS

A) Existen **cuatro entradas fijas**, es decir, que deben aparecer siempre:

- **FILENAME**

Especifica la ruta en la que se encuentra el archivo de la imagen que se quiere segmentar. Se escribe entre comillas simples. Por ejemplo:

```
seg ('../images/draft.jpg', margen, flag_marco, color_option)
```

- **MARGEN**

Este parámetro es muy importante, ya que está directamente relacionado con el tiempo que tarde el proceso de segmentación y con la calidad de la misma. Como ya se adelantó en el capítulo anterior el proceso de segmentación se realizará parcialmente con pequeñas subimágenes, y el ***margen*** determinará la distancia en la que se agrandan dichas subimágenes para evitar posibles contornos no detectados (véase la figura de abajo).

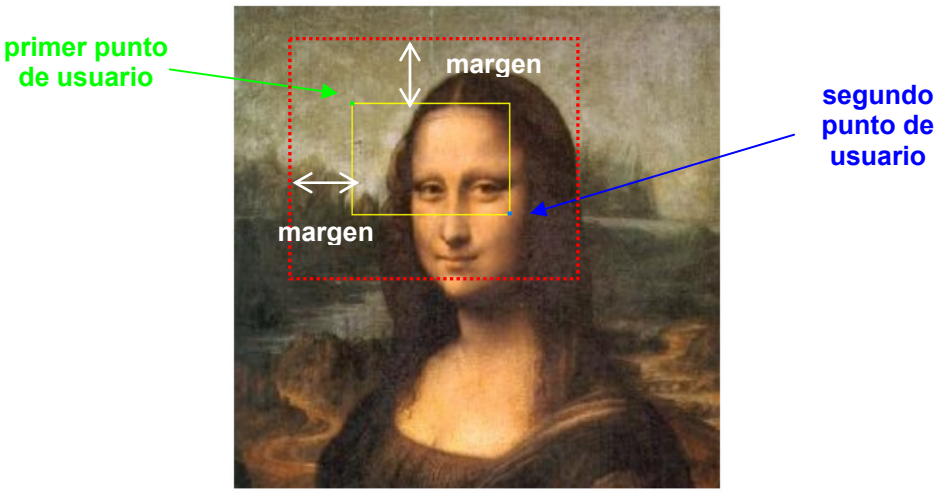


Figura 4.1. Explicación del parámetro de entrada margen.

- **FLAG_MARCO**

El ***marco*** de una imagen está compuesto por los píxeles pertenecientes a la primera y última fila, y a la primera y última columnas de ésta, es decir, determinará sus límites.

Figura 4.2. Representación del marco de una imagen (en rojo).

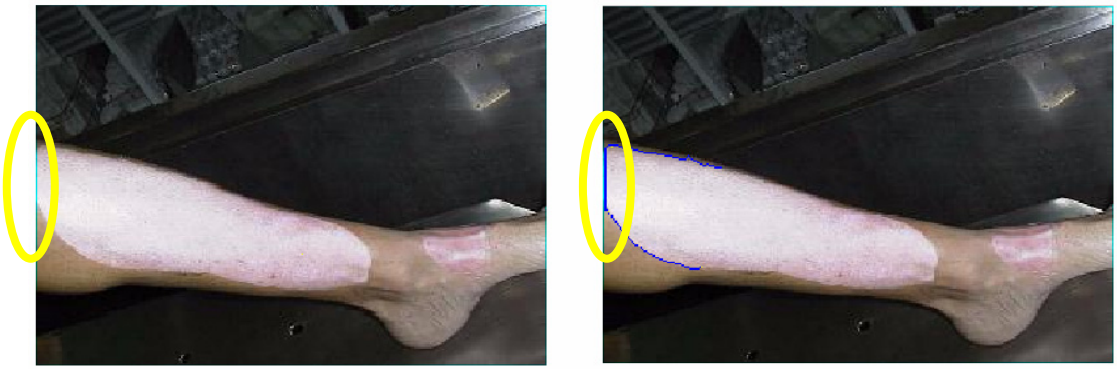
Este parámetro binario tiene los siguientes significados:

- Si flag_marco = 1 ⟹ el marco será considerado un camino mínimo

- Si flag_marco = 0 ⟹ el marco **NO** será considerado un camino mínimo

Cuando se activa este parámetro, *seg* llamará a la función '*marco_minimo*' (Apéndice A), la cual hará que todo el marco de la imagen sea un camino mínimo, esto no es más que poner todos los píxeles pertenecientes al marco a 0. De forma que, posteriormente, adquiera un coste ínfimo y pueda ser utilizado como un camino muy probable de formar parte de la segmentación.

Aunque la utilización de este parámetro no está tan clara a simple vista, es muy útil cuando las imágenes con las se trabajan están "cortadas", es decir, que el objeto que interesa segmentar no se encuentra completo. En esta situación se hace necesario que el usuario active la bandera del marco para que el programa tenga conocimiento del deseo de que la segmentación vaya por el marco cuando éste pinche el par de puntos en el marco, o muy cerca de él.



a) **b)**

Figura 4.3. a) Imagen donde el objeto a segmentar no está completo. **b)** Segmentación de la imagen anterior en la que se observa que uno de los segmentos es parte del marco.

- **COLOR_OPTION**

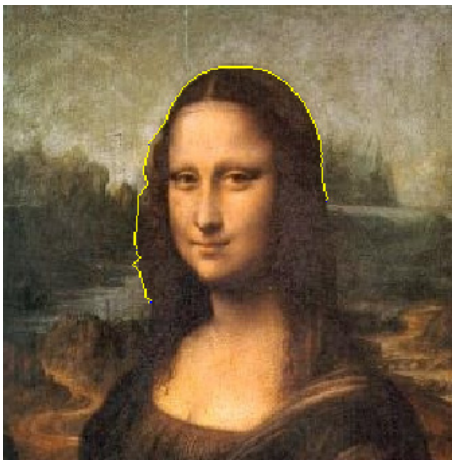
Su funcionalidad es simplemente indicar los colores con los que se presentan los píxeles marcados y los del camino de segmentación. Para ello el programa hace una llamada a la función '*colores*' (Apéndice A) que asigna los valores RGB que correspondan en cada caso. Es importante para la correcta visualización del proceso, por lo que se recomienda pensar antes que colores se ven mejor dependiendo de los que predominen en la imagen a segmentar.

En la siguiente tabla se muestran las diferentes opciones de color:

color_option	color segmento	color puntos pinchados
'b&y'	azul	amarillo
'y&b'	amarillo	azul
'r&g'	rojo	verde
'g&r'	verde	rojo
'b&w'	negro	blanco
'w&y'	blanco	negro
'w&g'	blanco	verde

Tabla 4.1. Opciones de colores en la presentación de una imagen segmentada.

a)



b)

c)

Figura 4.4. Imágenes segmentadas o semisegmentadas que señalan la importancia del color escogido.

a) color option 'r&g'.
b) color option 'y&b'.
c) color option 'w&g'.

B) Aparte de los cuatro argumentos fijos que tiene *seg*, las otras entradas que el programa recibe son los puntos que el usuario pincha con el ratón. Esta **intervención del usuario** es la causa principal para que la segmentación sea clasificada como semiautomática.

Nada más iniciarse el programa, y con todos los parámetros necesarios que éste necesita, se le pide al usuario que pinche sobre un punto inicial (píxel semilla), y seguidamente que pinche sobre otro segundo punto. El programa representará la frontera existente en el objeto recorriendo el camino entre los dos puntos indicados. A continuación se pide otro punto; el último punto dado anteriormente se convertirá en el nuevo píxel semilla y se representa el segmento entre esos dos píxeles.

Como se observa, el programa va pidiendo puntos al usuario y dibuja el segmento existente entre cada par de ellos, de forma que la segmentación se realiza segmento a segmento. El proceso de petición de puntos se va repitiendo hasta que el usuario pinche el segundo botón del ratón, momento en el que se da por concluida la segmentación.

Es obvio pensar que el usuario debe colocar el cursor del ratón lo más cercano al contorno del objeto que se quiera segmentar para obtener un buen resultado y evitar que la segmentación no sea la correcta (sobre todo en zonas con grandes diferencias de color muy próximas entre sí).

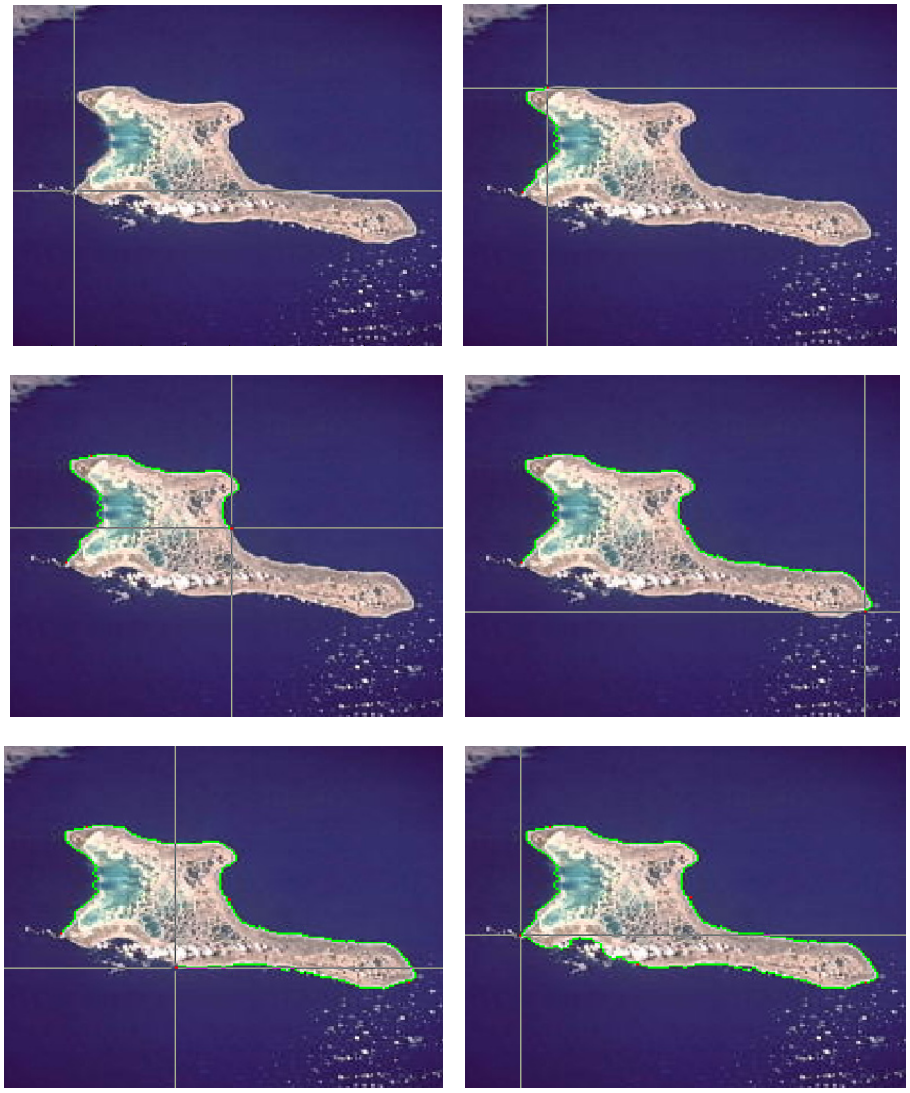


Figura 4.5. Representación de cada uno de los 'segmentos Live Wire' formados en el proceso de segmentación de una imagen.

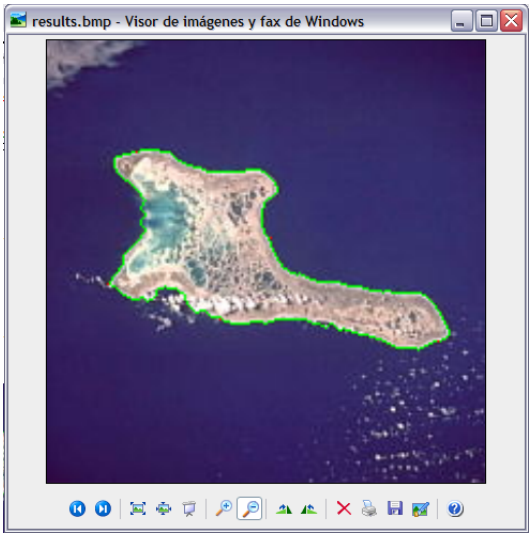
4.1.2. SALIDAS

El programa crea dos archivos cuando finaliza llamando a la función '*crear_resultados*' (Apéndice A):

'results.bmp': archivo de imagen donde se puede observar la segmentación final del objeto.

'results.txt': archivo de texto donde se pueden consultar datos de interés que se han producido durante el proceso de segmentación.

- Fecha y hora de la creación de los archivos
- Ruta y nombre del archivo de la imagen original a segmentar
- Tamaño de la imagen original
- Margen introducido por el usuario a la entrada
- Número de puntos pinchados por el usuario en la segmentación
- Tiempo de segmentación (en segundos)



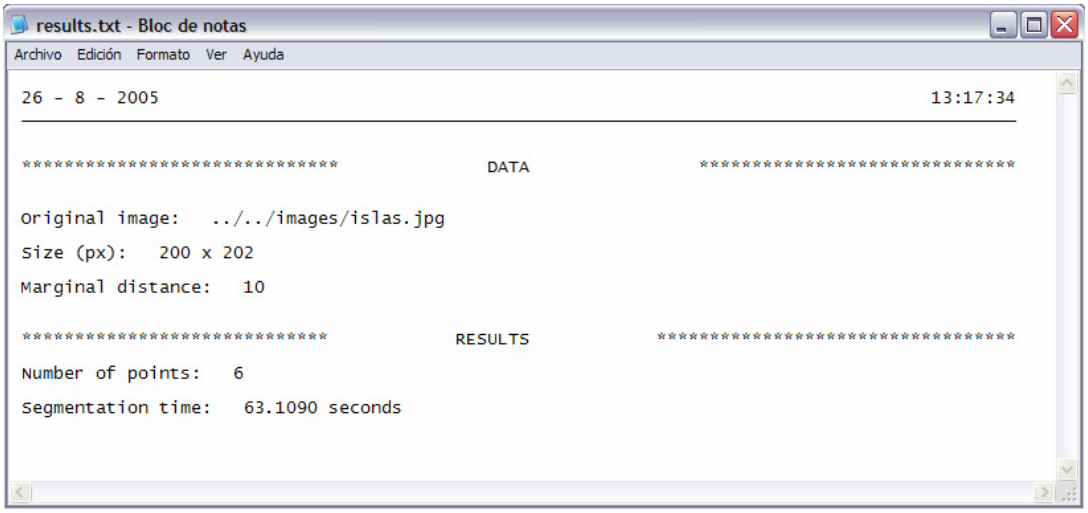


Figura 4.6. Archivos de salida, results.bmp y result.txt, de la segmentación de la imagen anterior.

4.2. BREVE DESCRIPCIÓN DE LAS FUNCIONES DISEÑADAS

A continuación se listan, por orden secuencial de aparición, las catorce funciones que forman la implementación en MATLAB del programa de segmentación de imágenes realizado. Los códigos se pueden consultar en el Apéndice A de este documento.

seg. Función principal del programa de implementación que recibe a la entrada la ruta del archivo con la imagen a segmentar, el margen que se aplicará a las subimágenes auxiliares, el indicador de si el marco tendrá coste mínimo o no y la combinación de colores con la que se representará la frontera del objeto.

colores. Función que recibe la opción de colores escogida por el usuario para representar la frontera del objeto segmentado y asigna los valores RGB a las etiquetas que proceda para que esto se lleve a cabo.

marco_minimo. Función que asigna a los píxeles pertenecientes al marco de la imagen, primera y ultima filas y columnas de la misma, un valor nulo para que se convierta en un conjunto de píxeles con coste local mínimo.

mcl. Función que recibe la imagen a segmentar a la entrada y crea la subimagen deseada devolviendo su mapa de coste local, así como las posiciones relativas a la subimagen de los píxeles semilla y posición. Para ello hace uso de las siguientes funciones:

> subimagen. Función que crea la subimagen limitada por el par de píxeles dado por el usuario.
>
> > comprueba. Función que comprueba que el tamaño de la subimagen creada con anterioridad no excede el de la imagen original y, en caso contrario, delimita la misma a las dimensiones iniciales.
>
> grad_mag. Función que calcula y devuelve la matriz magnitud del gradiente.
>
> grad_dir. Función que calcula y devuelve la matriz dirección del gradiente.

livewire. Función que crea el mapa de coste acumulado a partir del mapa de coste local y devuelve la matriz 'flechas' que contiene el conjunto de píxeles que forman el segmento Live Wire correspondiente al par de puntos pinchados por el usuario.

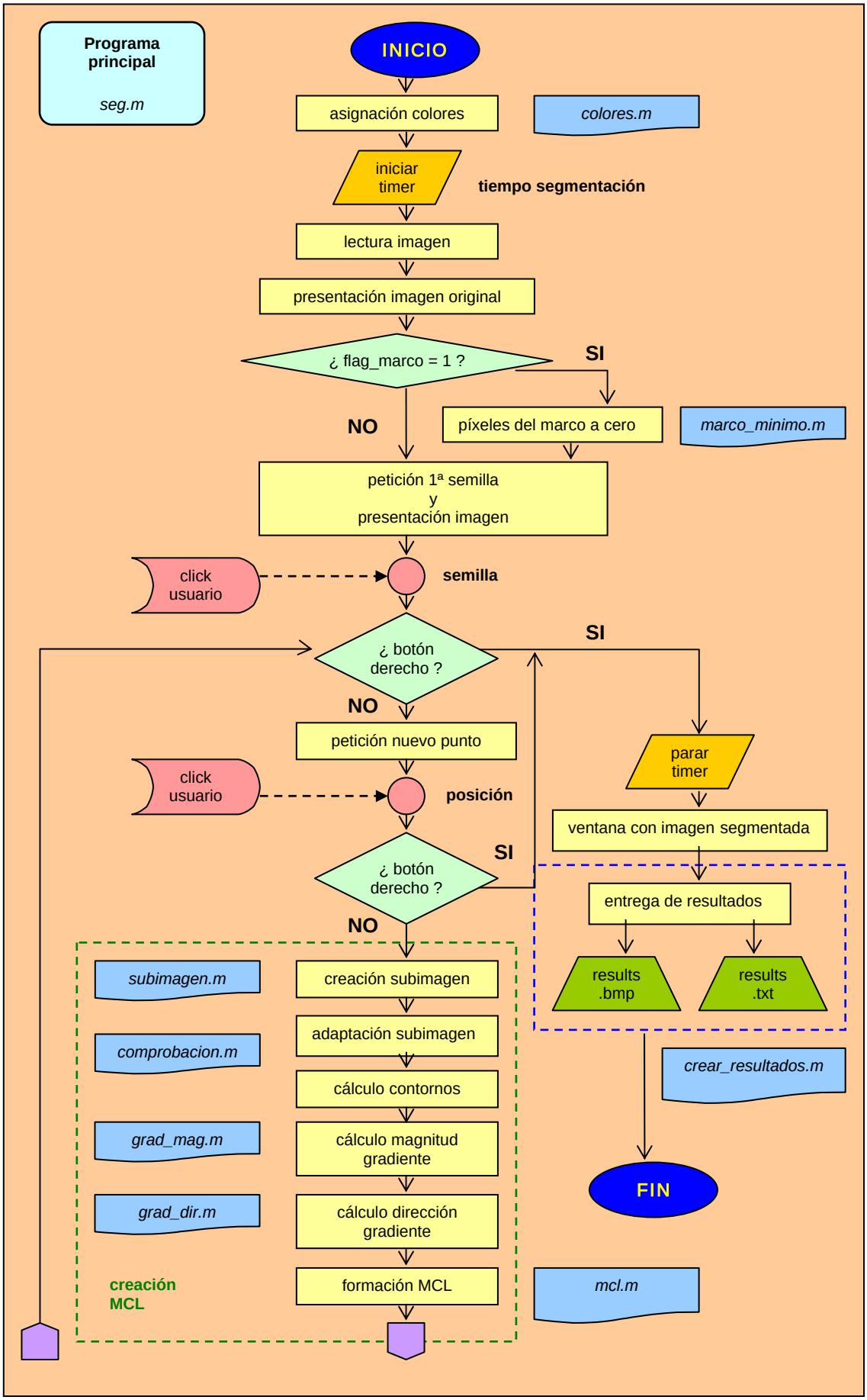
> ver_vecinos. Función que determina los vecinos de un píxel de la imagen.
>
> minimo. Función que devuelve el siguiente píxel a expandir (mínimo de la lista activa).

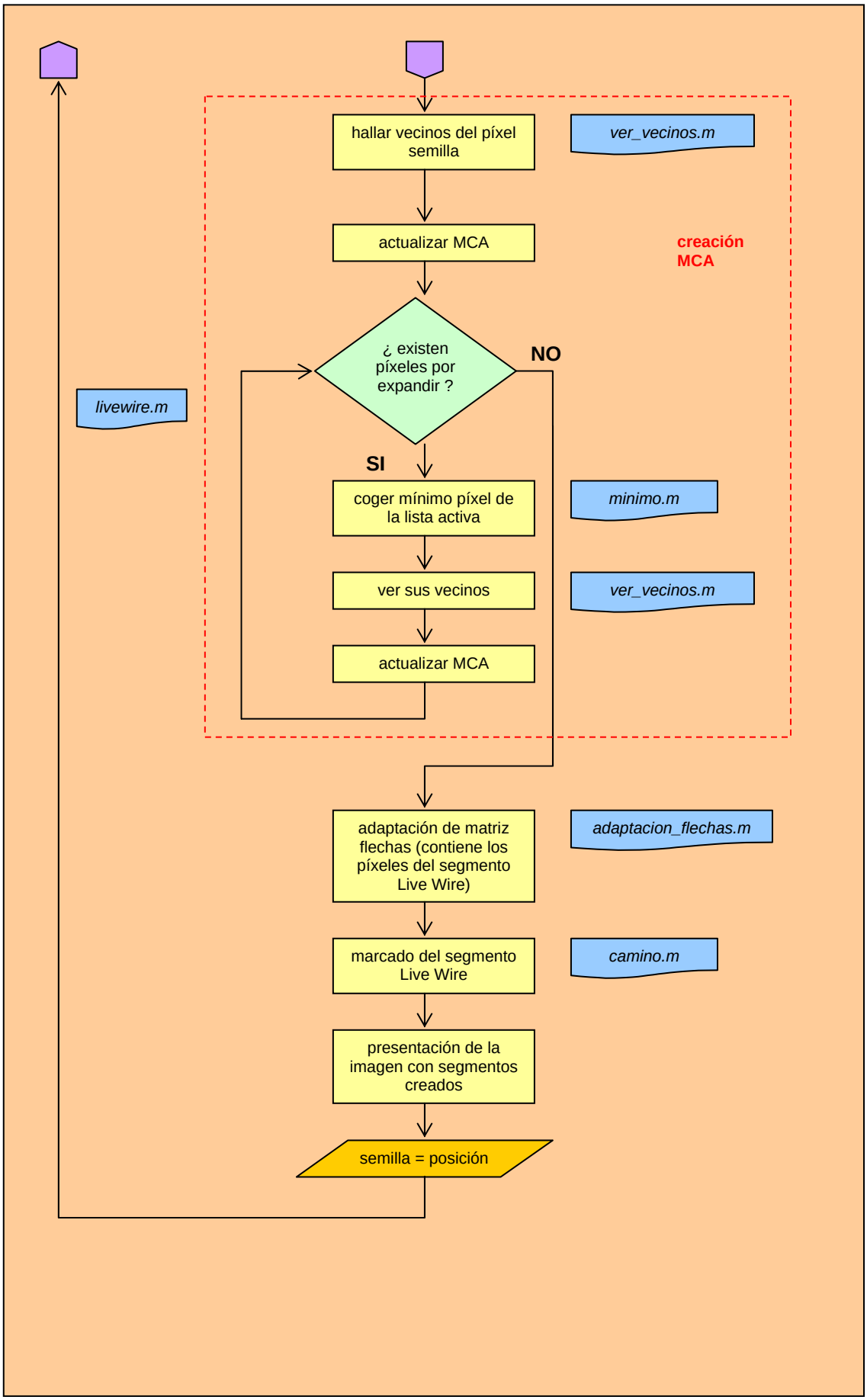
adaptación_flechas. Función auxiliar que copia la matriz flechas de la subimagen en la de la imagen original, facilitando posteriormente la representación del segmento Live Wire correspondiente.

camino. Función que recorre el segmento Live Wire marcándolo con el color elegido por el usuario y señalando también los píxeles que éste ha ido pinchando en el proceso de segmentación.

crear_resultados. Función que crea los archivos de salida 'results.bmp' y 'results.txt' con datos de interés del proceso de segmentación y la imagen segmentada respectivamente.

4.3. DIAGRAMA DE FLUJO COMPLETO




hallar vecinos del píxel semilla
ver_vecinos.m
actualizar MCA
creación MCA
¿ existen píxeles por expandir ?
NO
SI
livewire.m
coger mínimo píxel de la lista activa
minimo.m
ver sus vecinos
ver_vecinos.m
actualizar MCA
adaptación de matriz flechas (contiene los píxeles del segmento Live Wire)
adaptacion_flechas.m
marcado del segmento Live Wire
camino.m
presentación de la imagen con segmentos creados
semilla = posición